Matching...

deriv(U+V, X, DU + DV) :-
deriv(U, X, DU),
deriv(V, X, DV).

?- deriv(x^3 + x^2 + 1, x, D).
D = 1*3*x^2+1*2*x^1+0

- x^3 + x^2 + 1 matches U + V
  - x^3 + x^2 is bound to U
  - 1 is bound to V

Matching...

Can two terms A and F be “made identical,” by assigning values to their variables?

Two terms A and F match if
1. they are identical atoms
2. one or both are uninstantiated variables
3. they are terms $A = f_A(a_1, \ldots, a_n)$ and $F = f_F(f_1, \ldots, f_m)$, and
   (a) the arities are the same ($n = m$)
   (b) the functors are the same ($f_A = f_F$)
   (c) the arguments match ($a_i \equiv f_i$)

Matching

The rule:

deriv(U ^C, X, C * U ^L * DU) :-
number(C), L is C - 1,
deriv(U, X, DU).

?- deriv(x ^3, x, D).
D = 1*3*x^2

- x ^3 matches U ^C
  - x = U, C = 3
- x matches X
- D matches C * U ^L * DU
Matching...

Consequences of Prolog Matching:
- An uninstantiated variable will match any object.
- An integer or atom will match only itself.
- When two uninstatianted variables match, they *share*:
  - When one is instantiated, so is the other (with the same value).
- Backtracking undoes all variable bindings.

Matching Algorithm

FUNC Unify (A, F: term) : BOOL;
  IF Is_Var(F) THEN Instantiate F to A
  ELSIF Is_Var(A) THEN Instantiate A to F
  ELSIF Arity(F) \neq Arity(A) THEN RETURN FALSE
  ELSIF Functor(F) \neq Functor(A) THEN RETURN FALSE
  ELSE
    FOR each argument i DO
      IF NOT Unify(A(i), F(i)) THEN
        RETURN FALSE
    RETURN TRUE;

Matching – Example

<table>
<thead>
<tr>
<th>A</th>
<th>F</th>
<th>A \equiv F</th>
<th>variable subst.</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>sin((X))</td>
<td>sin(a)</td>
<td>yes (\theta = {X=a})</td>
<td></td>
</tr>
<tr>
<td>sin(a)</td>
<td>sin((X))</td>
<td>yes (\theta = {X=a})</td>
<td></td>
</tr>
<tr>
<td>cos((X))</td>
<td>sin(a)</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>sin((X))</td>
<td>sin(cos(a))</td>
<td>yes (\theta = {X=\cos(a)})</td>
<td></td>
</tr>
</tbody>
</table>

Matching – Example...

<table>
<thead>
<tr>
<th>A</th>
<th>F</th>
<th>A \equiv F</th>
<th>variable subst.</th>
</tr>
</thead>
<tbody>
<tr>
<td>likes(c, X)</td>
<td>likes(a, X)</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>likes(c, X)</td>
<td>likes(c, Y)</td>
<td>yes (\theta = {X=Y})</td>
<td></td>
</tr>
<tr>
<td>likes(X, X)</td>
<td>likes(c, Y)</td>
<td>yes (\theta = {X=c, X=Y})</td>
<td></td>
</tr>
<tr>
<td>likes(X, X)</td>
<td>likes(c, _X_47)</td>
<td>yes (\theta = {X=c, X=.47})</td>
<td></td>
</tr>
<tr>
<td>likes(c, a(X))</td>
<td>likes(V, Z)</td>
<td>yes (\theta = {V=c, Z=a(X)})</td>
<td></td>
</tr>
<tr>
<td>likes(X, a(X))</td>
<td>likes(c, Z)</td>
<td>yes (\theta = {X=c, Z=a(X)})</td>
<td></td>
</tr>
</tbody>
</table>
Matching – Example...

- The second step is to try to unify the first argument of the goal (X) with the first argument of the clause head (A).
- They are both variables, so that works OK.
- From now on A and X will be treated as identical (they are in the list of variable substitutions θ).

Matching – Example

- From Prolog for Programmers, Kluzniak & Szpakowicz, page 18.
- Assume that during the course of a program we attempt to match the goal \( p(X, b(X, Y)) \) with a clause \( C \), whose head is \( p(X, b(X, Y)) \).
- First we’ll compare the arity and name of the functors. For both the goal and the clause they are 2 and \( p \), respectively.

Slide 13-8

Slide 13-9

\[
p(X, b(X, Y))
\]

\[
p([A], b(c, A)) :- \ldots
\]

\[\theta = \{A = X\}\]

\[
p(A, b(c, A)) :- \ldots
\]
Matching – Example...

• Finally, we match A and Y. Since A=X and X=c, this means that Y=c as well.

Slide 13–12

• Next we try to match the second argument of the goal (b(X, Y)) with the second argument of the clause head (b(c, A)).

• The arities and the functors are the same, so we go on to try to match the arguments.

• The first argument in the goal is X, which is matched by the first argument in the clause head (c). I.e., x and c are now treated as identical.

Slide 13–13

p(X, b(X, Y))

p(A, b(c, A)) :- ...  
θ = {A = X, X = c, A = Y}

Slide 13–14

p(X, b(c, A)) :- ...  
θ = {A = X, X = c}

Slide 13–15
## Executing Prolog

EXECUTE

\[
\text{FUNC } \text{Execute} \ (G = G_1, G_2, \ldots, G_m; \ \text{Result}) = \begin{cases} 
\text{Result} := \text{Yes} & \text{if } \text{Is}_\text{Empty}(G) \ \text{THEN} \ \\
\text{Result} := \text{No} & \text{ELSE} \end{cases}
\]

\[
\text{WHILE } \text{Result} = \text{No} \ \& \ i \leq \text{NoOfClauses} \ \text{DO} \\
\quad \text{Clause} := H_i := B_1, \ldots, B_n; \\
\quad \text{IF } \text{Unify}(G_1, \text{Clause}, \ \theta) \ \text{THEN} \ \\
\quad \quad G' := \text{substitute}(B_1, \ldots, B_n, G_2, \ldots, G_m, \ \theta); \\
\quad \quad \text{Execute}(G', \ \text{Result}); \\
\quad \quad \text{ENDIF} \ \\
\quad i := i + 1; \\
\text{ENDDO ENDIF}
\]

---

## Execution Example

% From the Northern Exposure FAQ
% friend(of, kind(name, regular)).
friend(maggie, person(eve, yes)).
friend(maggie, moose(morty, yes)).
friend(maggie, person(harry, no)).
friend(maggie, person(bruce, no)).
friend(maggie, person(glenn, no)).
friend(maggie, person(dave, no)).
friend(maggie, person(rick, no)).
friend(maggie, person(mike, yes)).
friend(maggie, person(joel, yes)).

cause_of_death(morty, copper_deficiency).
cause_of_death(harry, potato_salad).
cause_of_death(bruce, fishing_accident).
cause_of_death(glenn, missile).
cause_of_death(dave, hypothermia).
cause_of_death(rick, hit_by_satellite).
cause_of_death(mike, none_yet).
cause_of_death(joel, none_yet).

male(morty). male(harry). male(bruce).
male(glenn). male(dave). male(rick).
male(mike). male(joel). female(eve).

---

## Executing Prolog

EXECUTE

\[
\text{FUNC } \text{Execute} \ (G = G_1, G_2, \ldots, G_m; \ \text{Result}) = \begin{cases} 
\text{Result} := \text{Yes} & \text{if } \text{Is}_\text{Empty}(G) \ \text{THEN} \ \\
\text{Result} := \text{No} & \text{ELSE} \end{cases}
\]

\[
\text{WHILE } \text{Result} = \text{No} \ \& \ i \leq \text{NoOfClauses} \ \text{DO} \\
\quad \text{Clause} := H_i := B_1, \ldots, B_n; \\
\quad \text{IF } \text{Unify}(G_1, \text{Clause}, \ \theta) \ \text{THEN} \ \\
\quad \quad G' := \text{substitute}(B_1, \ldots, B_n, G_2, \ldots, G_m, \ \theta); \\
\quad \quad \text{Execute}(G', \ \text{Result}); \\
\quad \quad \text{ENDIF} \ \\
\quad i := i + 1; \\
\text{ENDDO ENDIF}
\]
alive(X) :- cause_of_death(X, none_yet).
pastime(eve, hypochondria).
pastime(mike, hypochondria).
pastime(X, golf) :- job(X, doctor).

job(mike, lawyer). job(adam, chef).
job(maggie, pilot). job(joel, doctor).

?- friend(maggie, person(B, yes)),
   male(B),
   alive(B),
   pastime(B, golf).

G1

\[ \theta = \{ B=mike \} \]

Replace G1 by <empty>
Substitute vars from \( \theta \)
male(mike), alive(mike),
pastime(mike, golf).

Slide 13–20

G1

\[ \theta = \{ B=eve \} \]

Replace G1 by <empty>
Substitute vars from \( \theta \)
male(eve), alive(eve),
pastime(eve, golf).

Slide 13–21

G1

\[ \theta = \{ B=mike \} \]

Replace G1 by <empty>
Substitute vars from \( \theta \)
male(mike), alive(mike),
pastime(mike, golf).

Slide 13–22

G1

\[ \theta = \{ B=eve \} \]

Replace G1 by <empty>
Substitute vars from \( \theta \)
male(eve), alive(eve),
pastime(eve, golf).
Execution Example...

- We skip a step here.
- pastime(mike, golf) unifies with pastime(X, golf) :- job(X, doctor).

  However, job(mike, doctor) fails, and we backtrack all the way up to the original query.

\[ Hi \leftarrow X_1, \ldots, X_n \]
\[ \theta = \{X = \text{joel}\} \]
Prolog So Far...

- A term is either a
  - a constant (an atom or integer)
  - a variable
  - a structure
- Two terms match if
  - there exists a variable substitution \( \theta \) which makes the terms identical.
- Once a variable becomes instantiated, it stays instantiated.
- Backtracking undoes variable instantiations.
- Prolog searches the database sequentially (from top to bottom) until a matching clause is found.